**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
ON APPEAL FROM THE EXAMINER TO THE BOARD OF PATENT APPEALS
AND INTERFERENCES**

In re Application of:	Antony John Rogers et al.
Serial No.	09/905,532
Filing Date:	July 14, 2001
Group Art Unit:	2437
Confirmation No.:	3485
Examiner:	Michael J. Pyzocha
Title:	***Detection of Viral Code Using Emulation of Operating System Functions***

Mail Stop: Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Reply Brief

Appellants respectfully submit this Reply Brief under C.F.R. § 41.41 in response to the Examiner's Answer sent March 11, 2010 (the "Examiner's Answer"). Appellants filed an Appeal Brief on November 4, 2009.

Appellants respectfully request the Board of Patent Appeals and Interferences (the "Board") to reverse these final rejections and instruct the Examiner to issue a Notice of Allowance with respect to these claims. In response to the Examiner's Answer, Appellants submit herewith this Brief in reply.

Amended Grounds of Rejection to be Reviewed on Appeal

1. Are Claims 1, 4, 8-10, 12-16, 20, 22, and 23 allowable under 35 U.S.C. § 103(a) over U.S. Patent No. 6,192,512 issued to Chess ("*Chess*") in view of U.S. Patent No. 5,851,057 issued to Nachenberg ("*Nachenberg*")?

2. Is Claim 21 allowable under 35 U.S.C. § 103(a) over 35 U.S.C. 103(a) as being unpatentable over *Chess* and *Nachenberg* in view of U.S. Patent No. 5,398,196 to Chambers ("*Chambers*")?

Arguments

In response to the Examiner's arguments in the Examiner's Answer, Appellants present the following:

I. Claim 11

Claim 11 stands rejected under 35 U.S.C. § 103(a) as being unpatentable over *Chess* in view of *Nachenberg*. In order to advance prosecution, Appellants have withdrawn the presentation of Claim 11 for review by the Board. In light of this withdrawal, Appellants present an Amended Grounds of Rejection to be Reviewed on Appeal Section above, and an Amended Pending Claims Section in Appendix A of the Reply Brief.

II. Issue 1: Claims 1, 4, 8-10, 12-16, 20, 22, and 23 are patentable over *Chess* in view of *Nachenberg* under 35 U.S.C. § 103(a)

Claims 1, 4, 8-10, 12-16, 20, 22, and 23 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over *Chess* in view of *Nachenberg*. Appellants respectfully submit that the proposed combination of *Chess* and *Nachenberg* fails to disclose these claims. These rejections are therefore improper and should be reversed by the Board.

A. *Chess*

Chess discloses a system that executes a source program in a virtual environment. *Chess*, Col. 4, Lines 33-49. During the execution of the source program, if the executed source program "attempts to access a virtualized mass storage medium . . . then the external program can be informed of the potentially viral nature of the source program." *Chess*, Col. 4, Lines 49-54. That is, *Chess* discloses a system that determines which source program to execute in a virtual environment, and then determines that the source program is potentially viral if the executed source program attempts to access a virtualized mass storage medium.

B. *Nachenberg*

Nachenberg discloses a system that "can detect viruses in . . . files having multiple entry points without requiring a prohibitively large amount of processing time." *Nachenberg*, Col. 2, Lines 47-50. According to *Nachenberg*, before a file is emulated, "P-code" is used to determine "which entry points [of a file] should be posted 518 for emulating." *Nachenberg*, Col. 8, Lines 55-57. After, the P-code determines which entry points should be posted for emulating, "the emulating module 426 emulates 534 the code at the entry point posted by the P-code" in order to determine whether the file is infected. *Nachenberg*, Col. 9, Lines 49-52 and 62-63. That is, in *Nachenberg* each entry point of a file is determined before an emulation ever occurs.

C. The proposed *Chess-Nachenberg* combination fails to disclose "in response to detecting an attempt to access the artificial memory region, determining an export table entry in the export table of the dynamically-linked library that is associated with the attempt to access the artificial memory region"

The Examiner's Answer alleges that the *Chess-Nachenberg* combination discloses "in response to detecting an attempt to access the artificial memory region, determining an export table entry in the export table of the dynamically-linked library that is associated with the attempt to access the artificial memory region" of Claim 1. In particular, the Examiner's Answer alleges that *Chess* discloses "in response to detecting an attempt to access the artificial memory region, determining a ***source program*** that is associated with the attempt to access the artificial memory region." *See Examiner's Answer*, Page 3 (emphasis added). Furthermore, the Examiner's Answer alleges that when *Chess* is "combined with Nachenberg, this source program being tested has a specific entry point or entry points (i.e. export table entry) being tested and which entry point that was tested must also be determined in order to accurately report that a file at an entry point is potentially viral." *Examiner's Answer*, Pages 7-8. Appellants, however, respectfully disagree.

In particular, the combination relied upon in the Examiner's Answer as allegedly disclosing the above limitations of Claim 1 is based entirely on the idea that *Chess* discloses "in response to detecting an attempt to access the artificial memory region, determining a ***source program*** that is associated with the attempt to access the artificial memory region." *See Examiner's Answer*, Pages 3 and 7. Appellants respectfully submit that *Chess* fails to make such a disclosure--and therefore, the *Chess-Nachenberg* combination fails to disclose the limitations of Claim 1.

First, the Examiner's Answer points to Col. 4, Lines 49-54 of Chess as disclosing "in response to detecting an attempt to access the artificial memory region, determining a ***source program*** that is associated with the attempt to access the artificial memory region." This relied upon passage, however, discloses nothing about determining a source program in response to detecting an attempt to access the artificial memory region. Instead, the passage clearly discloses that the source program is already being executed (which means that the source program has already been determined long before) in order to decide whether the executed source program is infected with a virus:

> By example, if it is found that ***the source program*** unexpectedly attempts to access a virtualized mass storage medium, and/or to create one or more copies of itself in a region of virtualized memory, or in a virtual file, then the external program can be informed of the potentially viral nature ***of the source program***.

Chess, Col. 4, Lines 49-54 (emphasis added). Furthermore, not only does the passage fail to make the alleged disclosure, but the alleged disclosure is completely inconsistent with the passage of *Chess*. For example, since ***the source program*** that is being executed is ***the source program*** that is attempting to access the virtualized memory, it makes absolutely no sense for *Chess* to make a determination about which source program is associated with the attempt to access the artificial memory--it is clearly the source program that is being executed. No determination is necessary.

Accordingly, the passage relied upon in the Examiner's Answer fails to disclose "in response to detecting an attempt to access the artificial memory region, determining a ***source program*** that is associated with the attempt to access the artificial memory region"--and therefore, the *Chess-Nachenberg* combination fails to disclose the limitations of Claim 1.

Second, the Examiner's Answer then states that *Chess* makes this alleged disclosure because "[i]n order for the reporting to occur the system must determine which one of the 'new' or 'suspected' files has been emulated to accurately report to the external program that the file is potentially viral." *Examiner's Answer*, Page 7. Appellants, however, note that the Examiner's Answer ***provides no support for such a statement***. Furthermore, the statement is inconsistent with *Chess*. For example, as is discussed above, *Chess* discloses that ***the source program*** that is attempting to access the virtualized memory, is ***the source program*** that is currently being executed--thus, eliminating any need for a determination to be made:

> By example, if it is found that ***the source program*** unexpectedly ***attempts*** to access a virtualized mass storage medium, and/or to create one or more copies of itself in a region of virtualized memory, or in a virtual file, then the external

> program can be informed of the potentially viral nature ***of the source program***.

Chess, Col. 4, Lines 49-54 (emphasis added). Appellants respectfully point to the present tense language, "the source program unexpectedly attempts," of the above passage as proof that the source program is currently being executed by *Chess*--thus, no determination is needed. In addition, *Chess* provides further proof that the source program that is associated with the attempt to access the artificial memory is the source program currently being executed--thus, eliminating the need for any determination:

> In general, during ***each step of the simulation*** the application program subsystem 100 determines what interaction(s) the macro has had with the simulated environment 130, and provides that information to the caller of the API 110, either via callbacks as simulation proceeds (step 235), or via a return code or other report when the simulation terminates (step 245).

Chess, Col. 6, Lines 50-56 (emphasis added). Since the external program is informed of the status of the source program (or macro) at each step of the execution of the source program, no determination needs to be made about which source program is associated with the attempt to access the artificial memory--it is clearly the source program currently being executed.

Accordingly, contrary to the Examiner's Answer, *Chess* fails to disclose "in response to detecting an attempt to access the artificial memory region, determining a ***source program*** that is associated with the attempt to access the artificial memory region." Since *Chess* fails to make this alleged disclosure, the *Chess-Nachenberg* combination fails to disclose the limitations of Claim 1.

Additionally, not only does the *Chess-Nachenberg* combination fail to disclose the limitations of Claim 1, but *Nachenberg* fails to cure this deficiency. In particular, although *Nachenberg* discloses determining entry points of a file, *Nachenberg* fails to disclose that these determinations are made "in response to detecting an attempt to access the artificial memory region," as is recited in Claim 1. Accordingly, the *Chess-Nachenberg* combination fails to disclose the limitations of Claim 1.

For at least this reason, Appellants respectfully request the Board to reverse the rejection of Claim 1.

Claims 10, 12, and 14 each include limitations related to "in response to detecting an attempt to access the artificial memory region, determining an export table entry in the export table of the dynamically-linked library that is associated with the attempt to access the artificial memory region" Therefore, for at least the reasons discussed above with regard to Claim 1, Appellants respectfully request the Board to reverse the rejections of Claims 10, 12, and 14.

Claims 4, 8, 9, 13, 15, 16, 20, 21, 22, and 23 each depend, either directly or indirectly, from Claims 1, 12, and 14, respectively. Therefore, for at least the reasons discussed above with regard to Claims 1, 12, and 14, Appellants respectfully request the Board to reverse the rejections of Claims 4, 8, 9, 13, 15, 16, 20, 21, 22, and 23.

D. *Nachenberg* teaches away from the proposed *Chess-Nachenberg* combination

The Examiner's Answer alleges that the *Chess-Nachenberg* combination discloses "in response to detecting an attempt to access the artificial memory region, determining an export table entry in the export table of the dynamically-linked library that is associated with the attempt to access the artificial memory region" of Claim 1. Appellants, however, respectfully disagree.

Appellants respectfully submit that it is improper to combine *Chess* and *Nachenberg* to disclose these limitations of Claim 1 because *Nachenberg* teaches away from the *Chess-Nachenberg* combination. *See* M.P.E.P. § 2145(X)(D)(2) (citing *In re Grasselli*, 713 F.2d 731, 743, (Fed. Cir. 1983)) ("It is improper to combine references where the references teach away from their combination.). In particular, the Examiner's Answer alleges that the limitations of Claim 1 are disclosed by the *Chess-Nachenberg* combination because *Chess* discloses "in response to detecting an attempt to access the artificial memory region, determining a ***source program*** that is associated with the attempt to access the artificial memory region," and when *Chess* is "combined with Nachenberg, this source program being tested has a specific entry point or entry points (i.e. export table entry) being tested and which entry point that was tested must also be determined in order to accurately report that a file at an entry point is potentially viral." *See Examiner's Answer*, Pages 3 and 7-8 (emphasis added). This combination, however, is improper because *Nachenberg* expressly discloses that any determinations of an entry point of a file are ***made before the source program is ever executed***--thus clearly teaching away from *Chess*'s requirement that ***the source program be executed before*** a determination about the source program is made.

As an example, *Chess* discloses a system that executes a source program in a virtual environment. *Chess*, Col. 4, Lines 33-49. During the execution of the source program, if the executed source program "attempts to access a virtualized mass storage medium . . . then the external program can be informed of the potentially viral nature of the source program." *Chess*, Col. 4, Lines 49-54. Therefore, even if the Examiner's Answer is correct in alleging that *Chess* discloses "in response to detecting an attempt to access the artificial memory region, determining a ***source program*** that is associated with the attempt to access the artificial memory region" (which Appellants dispute, as is discussed above), then ***the source program must be executed before such a determination is ever made.***

Contrary to the requirement of *Chess* that the source program be executed before the determination about the source program is made, *Nachenberg* expressly discloses that any determinations of an entry point of a file are ***made before the file is ever executed***. *See Nachenberg*, Col. 8, Lines 55-57; Col. 9, Lines 49-52 and 62-63. In particular, *Nachenberg* expressly discloses that they must be made before execution because "[u]sing the P-code to ***preprocess*** regions of the file (100) and select only those regions or entry points that are likely to contain a virus for subsequent scanning and/or emulating allows the VDS (400) to examine files for viruses that infect places other than the main entry point in a ***reasonable amount of time***." *Nachenberg*, Col. 3, Lines 23-28 (emphasis added). Thus, *Nachenberg* expressly discloses that any determinations of an entry point of a file must be made before the file is ever executed.

Since *Nachenberg* expressly discloses that any determinations of an entry point of a source program are ***made before the source program is ever executed***, *Nachenberg* clearly teaches away from *Chess*'s requirement that ***the source program be executed before*** a determination about the source program is made. Accordingly, the proposed *Chess*-Nachenberg combination is improper.

For at least this reason, Appellants respectfully request the Board to reverse the rejection of Claim 1.

Claims 10, 12, and 14 each include limitations related to "in response to detecting an attempt to access the artificial memory region, determining an export table entry in the export table of the dynamically-linked library that is associated with the attempt to access the artificial memory region" Therefore, for at least the reasons discussed above with regard to Claim 1, Appellants respectfully request the Board to reverse the rejections of Claims 10, 12, and 14.

Claims 4, 8, 9, 13, 15, 16, 20, 21, 22, and 23 each depend, either directly or indirectly, from Claims 1, 12, and 14, respectively. Therefore, for at least the reasons discussed above with regard to Claims 1, 12, and 14, Appellants respectfully request the Board to reverse the rejections of Claims 4, 8, 9, 13, 15, 16, 20, 21, 22, and 23.

D. The proposed *Chess-Nachenberg* combination is based on impermissible hindsight

The Examiner's Answer alleges that the *Chess-Nachenberg* combination discloses "in response to detecting an attempt to access the artificial memory region, determining an export table entry in the export table of the dynamically-linked library that is associated with the attempt to access the artificial memory region" of Claim 1. Appellants, however, respectfully disagree.

Appellants respectfully submit that the proposed *Chess-Nachenberg* combination is based on impermissible hindsight, and is therefore improper. *See* M.P.E.P. § 2142 ("[I]mpermissible hindsight must be avoided and the legal conclusion must be reached on the basis of the facts gleaned from the prior art."). In particular, the Examiner's Answer alleges that the limitations of Claim 1 are disclosed by the *Chess-Nachenberg* combination because *Chess* discloses "in response to detecting an attempt to access the artificial memory region, determining a ***source program*** that is associated with the attempt to access the artificial memory region," and when *Chess* is "combined with Nachenberg, this source program being tested has a specific entry point or entry points (i.e. export table entry) being tested and which entry point that was tested must also be determined in order to accurately report that a file at an entry point is potentially viral." *See Examiner's Answer*, Pages 3 and 7-8 (emphasis added). This combination, however, is improper because it is based on impermissible hindsight.

In particular, contrary to the requirement of *Chess* that the source program be executed before the determination about the source program is made (as is discussed above), *Nachenberg* expressly discloses that any determinations of an entry point of a source program are ***made before the source program is ever executed*** (as is also discussed above). Despite these contradicting teachings, the Examiner's Answer alleges that *Chess* and *Nachenberg* can be combined in order to disclose "in response to detecting an attempt to access the artificial memory region, determining an export table entry in the export table of the dynamically-linked library that is associated with the attempt to access the artificial memory region" of Claim 1. Such an allegation, however, is clearly based on impermissible hindsight. For example, based on *Nachenberg's* disclosure that entry points of a file need to be determined before the file is emulated in order to "detect viruses in … files having multiple entry points without requiring a prohibitively large amount of processing time" (*see Nachenberg*, Col. 2, Lines 47-50), one of ordinary skill in the art would understand that the entry points should be determined before the emulation actually occurs. Contrary to this, however, the Examiner's Answer combines *Chess* and *Nachenberg* so that the determinations of entry points of a file are made after the file is actually executed--which is the exact opposite of everything *Nachenberg* teaches. Since the combination is clearly contrary to what *Nachenberg* teaches, the combination was clearly made using impermissible hindsight in order to allegedly disclose the limitations of Claim 1. As such, the proposed *Chess-Nachenberg* combination is improper.

For at least this reason, Appellants respectfully request the Board to reverse the rejection of Claim 1.

Claims 10, 12, and 14 each include limitations related to "in response to detecting an attempt to access the artificial memory region, determining an export table entry in the export table of the dynamically-linked library that is associated with the attempt to access the artificial memory region" Therefore, for at least the reasons discussed above with regard to Claim 1, Appellants respectfully request the Board to reverse the rejections of Claims 10, 12, and 14.

Claims 4, 8, 9, 13, 15, 16, 20, 21, 22, and 23 each depend, either directly or indirectly, from Claims 1, 12, and 14, respectively. Therefore, for at least the reasons discussed above with regard to Claims 1, 12, and 14, Appellants respectfully request the Board to reverse the rejections of Claims 4, 8, 9, 13, 15, 16, 20, 21, 22, and 23.

E. The Examiner's Answer's allegations of arguing against references individually are improper

The Examiner's Answer alleges that Appellants have argued against the references individually and asserts that "one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references." *Examiner's*

Answer, Pages 6 and 8. Even if the Examiner's Answer's point is true in some general sense, a rejection under 35 U.S.C. § 103 still requires that the cited references, either alone or in combination, disclose, teach, or suggest all the claim limitations. If the particular reference upon which the Examiner's Answer relies as allegedly disclosing a claim limitation does not teach that claim limitation, then plainly the reference has a deficiency and thus the proposed combination has a deficiency.

III. Issue 2: Claim 21 is patentable over *Chess* in view of *Nachenberg* and *Chambers* under 35 U.S.C. § 103(a)

The Examiner rejects Claim 21 under 35 U.S.C. § 103(a) as being unpatentable over *Chess* in view of *Nachenberg* and U.S. Patent No. 5,398,196 to Chambers ("*Chambers*"). Appellants respectfully submit that the proposed combination of *Chess*, *Nachenberg*, and *Chambers* fails to disclose this claim. This rejection is therefore improper and should be reversed by the Board.

Claim 21 depends from Claim 1. Therefore, for at least the reasons discussed above with regard to Claim 1, Appellants respectfully submit that the *Chess-Nachenberg* combination fails to disclose the limitations of Claim 21. Furthermore, *Chambers* does not cure this deficiency.

For at least this reason, Appellants respectfully request the Board to reverse the rejection of Claim 21.

Conclusion

Appellants have demonstrated that, for at least the foregoing reasons, the present invention, as claimed, is clearly patentable over the references cited by the Examiner. Therefore, Appellants respectfully request the Board to reverse the final rejection of the Examiner and instruct the Examiner to issue a Notice of Allowance of all pending claims.

Although Appellants believe no fees are due at this time in connection with this Reply Brief, the Commissioner is hereby authorized to charge any necessary fees and credit any overpayments to Deposit Account No. 02-0384 of BAKER BOTTS L.L.P.

Respectfully submitted,

BAKER BOTTS L.L.P.
Attorneys for Appellants

[illegible]

Luke K. Pedersen
Reg. No. 45,003
Tel. 214-953-6655

Date: 5-11-10

Customer Number: **05073**

APPENDIX A

Amended Pending Claims

1. A method of detecting viral code in subject files, comprising:
creating an artificial memory region spanning one or more components of the operating system, wherein the artificial memory region is associated with an export table of a dynamically-linked library;

emulating execution of at least a portion of computer executable code in a subject file;

monitoring attempts by the emulated computer executable code to access the artificial memory region;

in response to detecting an attempt to access the artificial memory region, determining an export table entry in the export table of the dynamically-linked library that is associated with the attempt to access the artificial memory region; and

determining based on the export table entry associated with the attempt to access the artificial memory region that the emulated computer executable code is viral.

4. The method of claim 1, further comprising:
emulating functionality of an identified operating system call while monitoring the operating system call to determine whether the computer executable code is viral.

8. The method of claim 1, further comprising monitoring access by the emulated computer executable code to dynamically linked functions.

9. The method of claim 8, wherein the artificial memory region spans a jump table containing pointers to the dynamically linked functions.

10. A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform method steps for detecting viral code in subject files, the method steps comprising:

creating an artificial memory region spanning one or more components of the operating system, wherein the artificial memory region is associated with an export table of a dynamically-linked library;

emulating execution of at least a portion of computer executable code in a subject file;

monitoring attempts by the emulated computer executable code to access the artificial memory region;

in response to detecting an attempt to access the artificial memory region, determining an export table entry in the export table of the dynamically-linked library that is associated with the attempt to access the artificial memory region; and

determining based on the export table entry associated with the attempt to access the artificial memory region that the emulated computer executable code is viral.

12. A computer data signal embodied in a computer-readable medium which embodies instructions executable by a computer for detecting in a subject file viral code that uses calls to an operating system, the signal comprising:

a first segment comprising CPU emulator code, wherein the CPU emulator code emulates execution of at least a portion of computer executable code in the subject file;

a second segment comprising memory manager code, wherein the memory manager code creates an artificial memory region spanning components of the operating system, wherein the artificial memory region is associated with an export table of a dynamically-linked library; and

a third segment comprising monitor code, wherein the monitor code:

monitors attempts by the emulated computer executable code to access the artificial memory region;

in response to detecting an attempt to access the artificial memory region, determining an export table entry in the export table of the dynamically-linked library that is associated with the attempt to access the artificial memory region;_and

determines based on the export table entry associated with the attempt to access the artificial memory region that the emulated computer executable code is viral.

13. The computer data signal of claim 12, further comprising:

a fourth segment comprising analyzer code, wherein the analyzer code emulates functionality of the identified operating system call to determine whether the computer executable code is viral.

14. An apparatus for detecting in a subject file viral code that uses calls to an operating system, comprising:

a processor;

a memory;

a CPU emulator;

a memory manager component that creates an artificial memory region spanning at least a portion of the memory associated with an export table of a dynamically-linked library; and

a monitor component, wherein the CPU emulator emulates execution of at least a portion of computer executable code in the subject file, and the monitor component:

monitors attempts by the emulated computer executable code to access the artificial memory region;

in response to detecting an attempt to access the artificial memory region, determining an export table entry in the export table of the dynamically-linked library that is associated with the attempt to access the artificial memory region;_and

determines based on the export table entry associated with the attempt to access the artificial memory region that the emulated computer executable code is viral.

15. The apparatus of claim 14, further comprising:

an auxiliary component; and

an analyzer component,

wherein the auxiliary component emulates functionalities of an identified operating system call, and the monitor component monitors the operating system call to determine whether the computer executable code is viral, while emulation continues.

16. The apparatus of claim 15, wherein the auxiliary component emulates functionalities of the operating system call.

20. The apparatus of claim 14, wherein the artificial memory region created by the memory manager component spans a jump table containing pointers to dynamically linked

functions, and the monitor component monitors access by the emulated computer executable code to the dynamically linked functions.

21. The method of claim 1, further comprising:

monitoring accesses by the emulated computer executable code to the artificial memory region to detect looping; and

determining based on a detection of looping that the emulated computer executable code is viral.

22. The method of claim 1, wherein creating an artificial memory region comprises creating a custom version of the export table with predetermined values for the entry points.

23. The method of claim 1, further comprising:

monitoring access by the emulated computer executable code to dynamically linked functions; and

determining based on attempted access to dynamically linked functions that the emulated computer executable code is viral.